What is claimed is:

1. A method for balancing the load of an n-dimensional array of processing elements, wherein each dimension of said array includes said processing elements arranged in a plurality of lines and wherein each of said processing elements has a local number of tasks associated therewith, the method comprising:

balancing at least one line of processing elements in a first dimension;

balancing at least one line of processing elements in a next dimension; and

repeating said balancing at least one line of processing elements in a next dimension for each dimension of said n-dimensional array.

2. The method of claim 1 wherein at least one of balancing at least one line of processing elements in a first dimension and balancing at least one line of PEs in a next dimension comprises:

calculating a total number of tasks for said line, wherein said total number of tasks for said line equals the sum of said local number of tasks for each processing elements on said line;

notifying each processing elements on said line of said total number of tasks for said line;

calculating a local mean number of tasks for each processing elements on said line;

calculating a local deviation for each processing elements on said line;

determining a first local cumulative deviation for each processing elements on said line;

determining a second local cumulative deviation for each processing elements on said line; and

redistributing tasks among said processing elements on said line in response to at least one of said first local cumulative deviation and said second local cumulative deviation.

3. The method of claim 2 wherein two or more lines in at least one of said first dimension and said next dimension are balanced in parallel.

4. The method of claim 2 wherein said calculating a total number of tasks for said line comprises sequentially summing said local number of tasks for each processing elements on said line from a first end of said line to a second end of said line.

5. The method of claim 2 wherein said calculating said total number of tasks for said line includes solving the equation $V = \sum_{i=0}^{i=N-1} v_i$, where V represents said total number of tasks for said line, N represents the number of processing elements on said line, and v_i represents said local number of tasks for a local PE_r on said line.

6. The method of claim 2 wherein said notifying step includes passing said total number of tasks from a second end of said line to a first end of said line.

7. The method of claim 2 wherein said calculating a local mean number of tasks includes solving the equation $M_r = Trunc((V + E_r)/N)$, where M_r represents said local mean for a local PE_r on said line, N represents the total number of PEs on said line, and E_r is a number in the range of 0 to $(N-1)$.

8. The method of claim 7 wherein each processing element has a different E_r value

9. The method of claim 7 wherein E_r controls said *Trunc* function such that said total number of tasks for said line is equal to the sum of the local mean number of tasks for each processing element on said line.

10. The method of claim 7 wherein said local mean $M_r = Trunc((V + E_r)/N)$ for each local PE_r on said line is equal to one of X and $(X+1)$.

11. The method of claim 2 wherein said calculating a local deviation for each processing element on said line includes finding a difference between said local number of tasks for each PE_r and said local mean number of tasks for each PE_r.

12. The method of claim 2 wherein said determining a first local cumulative deviation includes sequentially summing said local deviations for each PE_r from a first end of said line to an adjacent upstream PE_{r-1} on said line.

13. The method of claim 2 wherein said determining a second local cumulative deviation includes finding a difference between the negative of said local deviation for each PE_r and said first local cumulative deviation for each PE_r.

14. The method of claim 2 wherein said redistributing tasks among said processing elements on said line comprises:

transferring a task from a local PE_r to a left-adjacent PE_{r-1} if said first local cumulative deviation for said local PE_r is a negative value;

transferring a task from said local PE_r to a right-adjacent PE_{r+1} if said second local cumulative deviation for said local PE_r is a negative value.

15. The method of claim 2 wherein said redistributing tasks among said processing elements on said line comprises:

transferring a task from a local PE_r to a left-adjacent PE_{r-1} if said second local cumulative deviation for said local PE_r is a positive value;

transferring a task from said local PE_r to a right-adjacent PE_{r+1} if said first local cumulative deviation for said local PE_r is a positive value.

16. The method of claim 2 wherein said calculating a local mean number of tasks; said calculating a local deviation; said determining a first local cumulative deviation; said determining a second local cumulative deviation; and said redistributing tasks are completed in parallel for each processing element on said line.

17. The method of claim 16 wherein said calculating a local mean number of tasks; said calculating a local deviation; said determining a first local cumulative deviation; said determining a second local cumulative deviation; and said redistributing tasks are completed in parallel for each line in a selected dimension.

18. The method of claim 1 wherein said balancing each line of processing elements in a first dimension further comprises:

selecting one or more lines within said first dimension; and

shifting the number of tasks assigned to processing elements in said selected one or more lines.

19. The method of claim 2 wherein said calculating a local deviation, said determining a first local cumulative deviation, said determining a second local cumulative deviation, and said redistributing tasks among said processing elements are repeated until said local deviation, said first local cumulative deviation, and said second local cumulative deviation for each of said processing elements is zero.

20. A method for balancing an n-dimensional array of processing elements, wherein each of said n-dimensions is traversed by a plurality of lines and wherein each of said lines has a plurality of processing elements with a local number of tasks associated therewith, the method comprising:

balancing said plurality of lines in one dimension, wherein each of said balanced lines includes PEs with one of X local number of tasks and (X+1) local number of tasks;

substituting the value zero (0) for each processing element having X local number of tasks;

substituting the value one (1) for each processing element having (X+1) local number of tasks; and

shifting said values for each processing element within said balanced lines until a sum of said processing elements relative to a second dimension has only two values.

21. The method of claim 20 further comprising:

balancing said plurality of lines in a next higher dimension; and

repeating said balancing said plurality of lines in a next higher dimension for each remaining dimension of said n-dimensional array.

22. The method of claim 20 wherein said balancing said plurality of lines in one dimension comprises:

calculating a total number of tasks present within at least one of said lines;

notifying each processing element on said line of said total number of tasks for said line;

determining each processing element's share of said total number of tasks on said line;

determining a first local cumulative deviation for each processing element on said line;

determining a second local cumulative deviation for each processing element on said line;

redistributing tasks among each processing element on said line in response to at least one of said first local cumulative deviation and said second local cumulative deviation.

23. The method of claim 22 wherein said notifying each processing element comprises:

serially summing said total number of tasks present on said line; and

transmitting said total number of tasks to each processing element on said line.

24. The method of claim 22 wherein said determining each processing element's share of said total number of tasks comprises:

calculating a local mean number of tasks for each processing element on said line; and

calculating a local deviation from said local mean number of tasks for each processing element on said line by finding the difference between said local number of tasks and said local mean number of tasks for each processing element on said line.

25. The method of claim 24 wherein said calculating a local mean number of tasks for each processing element on said line comprises using a rounding function $M_r = Trunc((V + E_r) / N)$, where M_r represents said local mean of a local PE_r, N represents the total number of processing elements on said line, and E_r represents a number in the range of 0 to $(N-1)$.

26. The method of claim 25 wherein E_r controls said *Trunc* function such that said total number of tasks for said line is equal to the sum of the local mean number of tasks for each of said processing elements in said line.

27. The method of claim 24 wherein said local mean $M_r = Trunc((V + E_r) / N)$ for each local processing element on said line is equal to one of X and (X+1).

28. The method of claim 22 wherein said determining a first local cumulative deviation for each processing element on said line includes summing said local deviations for each upstream processing element on said line.

29. The method of claim 22 wherein said determining a second local cumulative deviation for each processing element on said line includes finding the difference between the negative of said local deviation and said first local cumulative deviation for each processing element on said line.

30. The method of claim 22 wherein said redistributing tasks among each processing element on said line in response to at least one of said first local cumulative deviation and said second local cumulative deviation comprises:

transferring a task from a first processing element on said line to a second processing element on said line if said first local cumulative deviation for said first processing element is a negative value; and

transferring a task from said second processing element on said line to said first processing element on said line if said first local cumulative deviation for said second processing element is a positive value.

31. The method of claim 22 wherein said redistributing tasks among each processing element on said line in response to at least one of said first local cumulative deviation and said second local cumulative deviation comprises:

transferring a task to a first processing element on said line from a second processing element on said line if said second local cumulative deviation for said first processing element is a negative value; and

transferring a task to said second processing element on said line from said first processing element on said line if said second local cumulative deviation for said second processing element is a positive value.

32. The method of claim 24 wherein said calculating a local deviation, said determining a first local cumulative deviation, said determining a second local cumulative deviation, and said redistributing tasks among said processing elements are repeated until said local deviation, said first local cumulative deviation, and said second local cumulative deviation for each of said processing elements is zero.

33. A memory device carrying a set of instructions which, when executed, perform a method comprising:

balancing at least one line of processing elements in a first dimension;

balancing at least one line of processing elements in a next dimension; and

repeating said balancing at least one line of processing elements in a next dimension for each dimension of said n-dimensional array.